

**Yee &
Associates, P.C.**

4100 Alpha Road
Suite 1100
Dallas, Texas 75244

Main No. (972) 385-8777
Facsimile (972) 385-7766

**RECEIVED
CENTRAL FAX CENTER**

MAR 28 2005

Facsimile Cover Sheet

To: Commissioner for Patents for Examiner Douglas B. Blair Group Art Unit 2142	Facsimile No.: 703/872-9306
From: Michele Morrow Legal Assistant to Gerald H. Glanzman	No. of Pages Including Cover Sheet: 29
Message: Enclosed herewith: <ul style="list-style-type: none">• Transmittal Document; and• Appeal Brief.	
Re: Application No. 09/651,585 Attorney Docket No: JP9-1999-0175US1	
Date: Monday, March 28, 2005	
Please contact us at (972) 385-8777 if you do not receive all pages indicated above or experience any difficulty in receiving this facsimile.	<i>This Facsimile is intended only for the use of the addressee and, if the addressee is a client or their agent, contains privileged and confidential information. If you are not the intended recipient of this facsimile, you have received this facsimile inadvertently and in error. Any review, dissemination, distribution, or copying is strictly prohibited. If you received this facsimile in error, please notify us by telephone and return the facsimile to us immediately.</i>

**PLEASE CONFIRM RECEIPT OF THIS TRANSMISSION BY
FAXING A CONFIRMATION TO 972-385-7766.**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Inagaki et al.**

Serial No.: 09/651,585

Filed: August 29, 2000

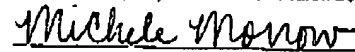
For: **Client Server System and Method
for Executing an Application Utilizing
Distributed Objects****36736**PATENT TRADEMARK OFFICE
CUSTOMER NUMBER§
§
§
§
§
§
§
§

Group Art Unit: 2142

Examiner: **Blair, Douglas B.**Attorney Docket No.: **JP9-1999-0175US1**Certificate of Transmission Under 37 C.F.R. § 1.8(a)

I hereby certify this correspondence is being transmitted via facsimile to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, facsimile number (703) 872-9306 on March 28, 2005.

By:


Michele MorrowTRANSMITTAL DOCUMENTCommissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

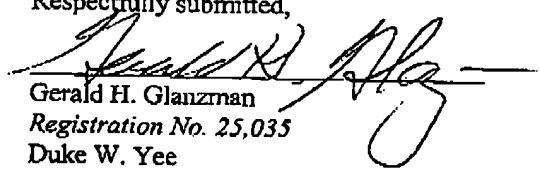
Sir:

ENCLOSED HERewith:

- Appeal Brief (37 C.F.R. 41.37).

A fee of \$500.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0461. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0461. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0461.

Respectfully submitted,


Gerald H. Glanzman

Registration No. 25,035

Duke W. Yee

Registration No. 34,285

YEE & ASSOCIATES, P.C.

P.O. Box 802333

Dallas, Texas 75380

(972) 385-8777

ATTORNEYS FOR APPLICANTS

RECEIVED
CENTRAL FAX CENTER

MAR 28 2005

Docket No. JP9-1999-0175US1

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Inagaki et al.

Serial No. 09/651,585

Filed: August 29, 2000

For: Client Server System and Method
for Executing an Application Utilizing
Distributed Objects§
§
§
§
§
§
§
§
§
§

Group Art Unit: 2142

Examiner: Blair, Douglas B.

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450Certificate of Transmission Under 37 C.F.R. § 1.8(a)

I hereby certify this correspondence is being transmitted via
facsimile to the Commissioner for Patents, P.O. Box 1450,
Alexandria, VA 22313-1450, facsimile number (703) 872-9306
on March 28, 2005.

By:



Michele Morrow

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on January 27, 2005.

The fees required under § 41.20(B)(2), and any required petition for extension of time for filing this brief and fees therefore, are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

(Appeal Brief Page 1 of 27)
Inagaki et al. - 09/651,585

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-15

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: NONE
2. Claims withdrawn from consideration but not canceled: NONE
3. Claims pending: 1-15
4. Claims allowed: NONE
5. Claims rejected: 1-15
6. Claims objected to: NONE

C. CLAIMS ON APPEAL

The claims on appeal are: 1-15

STATUS OF AMENDMENTS

A Response to Final Office Action was filed on December 29, 2004; however, no amendments were made to the claims in the Response. Therefore, claims 1-15 on appeal herein are as amended in the Response to Office Action filed June 28, 2004, and as rejected in the Final Office Action mailed November 3, 2004.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

The subject matter of claim 1 is directed to client server system that uses distributed objects (see page 12, lines 3-11 in conjunction with **Figure 1** and page 27, lines 10-18 in conjunction with **Figure 21**). A client 73 is connected to a communication network 74, 75 (see page 27, lines 13-18 and **Figure 21**) for performing an access request to an object. An application server 20 or 71 (see page 12, lines 6-7 in conjunction with **Figure 1** and page 27, lines 11-13 in conjunction with **Figure 21**) performs an application by an actual object 16 (see page 12, line 10 and **Figure 1**) according to the access request by the client (reference number 11 in **Figure 1** and reference number 73 in **Figure 21**). An object pool server 72 (see page 27, lines 13-15 and **Figure 21**) is connected to the client 73 through the communication network 74, 75 and is connected to the application server 71 for pooling a proxy object 17 (see page 12, lines 10-11 and **Figure 1**) corresponding to the actual object 16 and for holding actual object management information that is information on the actual object 16. The application server 71 notifies the object pool server 72 of an event according to a change in status of the application and the object pool server 72 automatically updates the actual object management information according to the notification of the event from the application server 71 (see, for example, page 21, lines 1-7 and **Figure 13**, steps 131 and 132).

B. CLAIM 3 - INDEPENDENT

The subject matter of claim 3 is directed to an object pool that uses distributed objects. The object pool 30 includes a client request analyzing unit 41 (see page 15, lines 1-3 and **Figure 5** for analyzing an access request to an object, and an object information storage unit 35 (see page 14, lines 14-16 and **Figure 5**) for storing object information at the termination process of the object pool. An object creating unit 42 (see page 15, lines 2-3 and **Figure 5**) creates an object at the starting process of the object pool according to the object information stored by the object information storage unit 35, and an object managing unit 32 (see page 14, lines 9-12 and **Figure**

5) pools the object created by the object creating unit 42 before accessing the object from the client.

C. CLAIM 5 – INDEPENDENT

The subject matter of claim 5 is directed to a client server system using distributed objects. The system includes an object pool 30 (see page 12, lines 8-10 in conjunction with Figure 1, and page 14, line 9 to page 15, line 15 in conjunction with Figure 5) connected through a communication network 74, 75 (see page 27, lines 13-18 and Figure 21) to a client 11, 73 (see page 12, lines 4-6 in conjunction with Figure 1, and page 27, line 15 in conjunction with Figure 21) which sends an access request to an object, and for pooling objects and managing object information. An application execution environment 20 (see page 12, lines 6 and 7 and Figure 1) is connected to the object pool 30 for executing an application according to the access request of the client 11, and notifies the object pool 30 of an event according to a status change of the application, wherein the object pool 30 updates the object information according to the event notification from the application execution environment 20.

D. CLAIM 8 – INDEPENDENT

The subject matter of claim 8 is directed to an object pooling method for pooling objects in advance on a specified server to execute an application in a distributed system. The object pooling method comprises the steps of storing object information when a process of the server is terminated, creating objects according to the object information when a process of the server is started, and pooling the created objects (see page 21, line 21 to page 23, line 5 and Figures 14 and 15).

E. CLAIM 10 – INDEPENDENT

The subject matter of claim 10 is directed to an object pooling method on an object pool which is connected through a communication network to a client which sends an access request to an object, and is connected to an application execution environment in which an actual object is executed. The method comprises the steps of pooling a proxy object corresponding to an actual object in the application execution environment, recognizing a change in status of the

actual object in the application execution environment, and updating actual object management information according to the recognized status change (see page 20, line 17 to page 21, line 20 and Figure 13).

F. CLAIM 11 – INDEPENDENT

The subject matter of claim 11 is directed to a storage medium in which a program to be executed by a computer is stored so that it can be read by an input unit of the computer. The program makes the computer execute a monitoring process for monitoring execution status of a project which executes an application utilizing distributed objects, an event creating process for creating an event according to execution status of the project monitored by the monitoring process (see page 13, lines 4-5 and Figure 2), and an event issuing process for issuing the event created by the event creating process to an object pool server connected through a network or the like (see page 13, lines 5-7 and Figure 2).

G. CLAIM 12 – INDEPENDENT

The subject matter of claim 12 is directed to a storage medium in which a program to be executed by a computer is stored so that it can be read by an input unit of the computer. The computer is connected through a network 74, 75 (see page 27, lines 13-18 and Figure 21) or the like to an application server 71 for executing an application utilizing distributed objects, and to a client which requests the execution of the application. The program makes the computer execute an object pooling process for pooling the proxy objects corresponding to actual objects 16 to be executed by the application server (see page 12, lines 10-11 and Figure 1), and an updating process for receiving an event issued from the application server 71 and updating management information on the actual object 16 according to the event (see page 13, lines 15-24 and Figure 3, and page 21, lines 1-7 and Figure 13, steps 131 and 132).

H. CLAIM 14 – INDEPENDENT

The subject matter of claim 14 is directed to a program sending apparatus. The apparatus comprises a storage unit 35 (see page 14, lines 14-16 and Figure 5) for storing a software product which makes a computer execute an event forming program for forming an event

according to a change in status of an application utilizing distributed objects, and an object pooling program for pooling objects according to the event formed by the event forming process, and a sending unit for reading out the program from the storage unit, and sending the software product (see, generally, page 30, line 14 to page 31, line 8).

I. CLAIM 15 – INDEPENDENT

The subject matter of claim 15 is directed to a program sending apparatus. The apparatus comprises a storage unit 35 (see page 14, line 14-16 and Figure 5) for storing a program which makes a computer execute an object pooling process for pooling, on a server, objects associated with execution of an application utilizing distributed objects, an information storing process for storing object information in the server, and a creation sequence determining process for determining a sequence of objects to be created according to the object information stored by the information storing process. A sending unit reads out the program from the storage unit, and sends the program (see, generally, page 30, line 14 to page 31, line 8).

GROUND OF REJECTION TO BE REVIEWED ON APPEAL

A. GROUND OF REJECTION (Claims 1-15)

Claims 1-15 stand rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,233,623 to Jeffords et al.

ARGUMENT

A. GROUND OF REJECTION (Claims 1-15)

The Examiner has rejected claims 1-15 under 35 U.S.C. § 102(e) as being anticipated by Jcffords et al. (U.S. Patent No. 6,233,623). This rejection is respectfully traversed.

A.1. Claims 1 and 2

In rejecting claim 1, the Examiner states as follows:

As to claim 1, Jeffords teaches a client server system using distributed objects, comprising: a client connected to a communication network for performing an access request to an object (col. 14, line 36-col. 15, line 17); an application server for performing an application by an actual object according to the access request by said client (col. 14, line 36-col. 15, line 17); and an object pool server connected to said client through said communication network and connected to said application server for pooling a proxy object corresponding to said actual object and for holding actual object management information that is part of said actual object (col. 14, line 36-col. 15, line 17) wherein said application server notifies said object pool server of an event according to a change in status of said application, and said object pool server automatically updates said actual object management information according to the notification of said event from said application server (col. 14, line 36-col. 15, line 17).

Final Office Action dated November 3, 2004, pages 2 and 3.

Claim 1 of the present application reads as follows:

1. A client server system using distributed objects, comprising:
 - a client connected to a communication network for performing an access request to an object;
 - an application server for performing an application by an actual object according to the access request by said client; and

an object pool server connected to said client through said communication network and connected to said application server for pooling a proxy object corresponding to said actual object and for holding actual object management information that is information on said actual object, wherein said application server notifies said object pool server of an event according to a change in status of said application, and said object pool server automatically updates said actual object management information according to the notification of said event from said application server.

A prior art reference anticipates a claimed invention under 35 U.S.C. § 102 only if every element of the claimed invention is identically shown in that single prior art reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of a claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983).

Jeffords et al. (hereinafter Jeffords) does not disclose a client server system using distributed objects that includes “an object pool server connected to said client through said communication network and connected to said application server for pooling a proxy object corresponding to said actual object and for holding actual object management information that is information on said actual object, wherein said application server notifies said object pool server of an event according to a change in status of said application, and said object pool server automatically updates said actual object management information according to the notification of said event from said application server”; and, accordingly, does not anticipate claim 1.

The Examiner refers to col. 14, line 36-col. 15, line 17 of Jeffords, as disclosing the subject matter of claim 1. Appellants respectfully disagree. Col. 14, line 36-col. 15, line 17 of Jeffords is as follows:

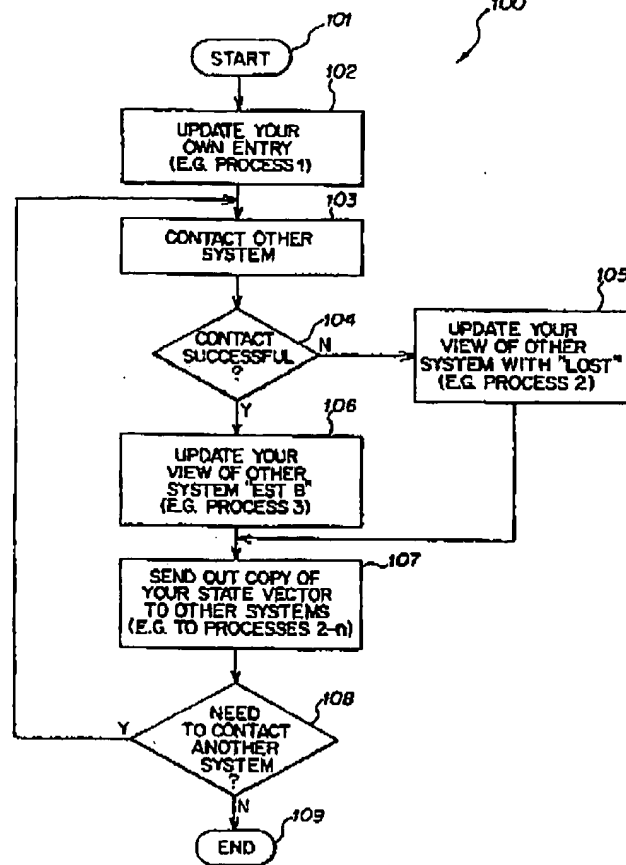
5. Updating State Information

Fig. 8 is a flowchart of a method for updating state information 100. Starting at Step 101, a particular system, (e.g., process 1) updates its own contact status vector (Step 102). Next, process 1 contacts the other systems (Step 103). Further action depends upon whether the contact is successful (Step 104); if not successful, process 1 updates its state vector with “lost” for the noncontacted system; e.g., process 2 (Step 105). If contact was

successful, process 1 updates its state vector with ESTB for the contacted system, e.g., process 3 (Step 106). In either case, process 1 then sends out a copy of its state vector to the other systems, e.g., processes 2-n (Step 107). Then process 1 continues to contact other systems (Step 108); when all processes have been attempted to be contacted, the method ends (Step 109).

The RRM has many areas of potential application. For example, in a switched communications network having a distributed call management system, a resource manager can be provided for each of a plurality of distributed call management processes. See, for example, the switched network described in copending and commonly owned U.S. Serial No. 08/188,238 filed Jan. 28, 1994 by K. Dobbins et al., hereby incorporated by reference in its entirety. A distributed memory space can be divided into separate pools for each of the following services: topology, directory, policy and call. A first benefit of incorporating RRM in this system is that every process is provided with a consistent view of the network. For example, every process can have access to a policy pool which may specify, for example, that node-a is permitted to communicate with node-b. As a further example, when a node in the network topology changes, it is important that every switch be notified of the change. Using RRM, state changes are seen every where so that every switch has the same view of the network topology. A second benefit is fault tolerance; for example, because the call pool has an object for every call, if one call management process goes down, another call management process can take over the calls in the call pool previously owned by the failed process; this can be done quickly with essentially no interruption in service. A third benefit is load sharing. The various call management processes can distribute ownership of calls in order to balance the workload. A fourth benefit is scalability; additional call processes can be added to the system easily without requiring reprogramming of the prior systems. Yet another important benefit from a programming perspective is that the programmer does not need to know where any given method or change of object is executed, only that the result is the same.

Figure 8 of Jeffords is reproduced below for the convenience of the Board:



Jeffords describes a procedure in which a particular system updates its own contact status vector and then contacts other systems (steps 102 and 103 in Figure 8 of Jeffords). If contact is made with another system, the particular system updates its state vector with ESTB (contact with the process has been established) for the contacted system, and then sends out a copy of its "state vector" to other systems (steps 106 and 107 in Figure 8 of Jeffords). The particular system then contacts other systems and steps 106 and 107 are repeated until all systems have been attempted to be contacted.

Jeffords defines a "state vector" in col. 12, lines 52-58 as follows:

A state vector is a one-dimensional associative array of logical object name to logical object state. Each name is an "index" into the vector, and each state is stored in a "slot" associated with an index. A state vector is generated by an object and describes what that object thinks the state of all objects in the vector is. It is the object's view of itself and "relative" view of all other objects.

Jeffords does not disclose, in col. 14, line 36-col. 15, line 17 or elsewhere in the patent, "an object pool server connected to said client through said communication network and connected to said application server for pooling a proxy object corresponding to said actual object and for holding actual object management information that is information on said actual object, wherein said application server notifies said object pool server of an event according to a change in status of said application", and that the object pool server "automatically updates said actual object management information according to the notification of said event from said application server". Jeffords, instead, appears to be concerned with contacting other systems in a network, and then updating its own state vector and sending a copy of the state vector to the other contacted systems.

In responding to Appellants' arguments presented in the Response to Office Action dated June 28, 2004, the Examiner states, in part:

Jeffords describes stored object information from the resources at col. 5, lines 37-49. The stored resource objects states are continually stored in the distributed memory and when the Replicated Resource Management objects are initialized they communicate with the resource objects.

Final Office Action dated November 3, 2004, page 5.

Col. 5, lines 37-49 of Jeffords is as follows:

Once the RRM subsystem is initialized with the above information, its services may be used by the application. At this point, the application may do the following:

- examine/use the resource objects in the resource pools
- instantiate new resource objects in the resource pools
- delete resource objects from the resource pools
- change the attributes of resource objects
- receive attribute change notifications for resource objects
- use object-level messaging and object-level remote procedure calls
- receive contact lost/established notifications

This recitation states only that following initialization of the RRM system, an application may perform various actions with respect to resource objects. The recitation is not a teaching that an "application server notifies said object pool server of an event according to a change in status of said application, and said object pool server automatically updates said actual object

management information according to the notification of said event from said application server” as recited in claim 1. Only the present application contains such a disclosure.

For at least all the above reasons, Jeffords does not anticipate claim 1, and claim 1 is believed to be allowable over Jeffords in its present form.

Claim 2 depends from and further restricts claim 1, and is also not anticipated by Jeffords, at least by virtue of its dependency.

A.2. Claims 3 and 4

Independent claim 3 is as follows:

3. An object pool using distributed objects, comprising:
 - a client request analyzing unit for analyzing an access request to an object;
 - an object information storage unit for storing object information at the termination process of said object pool;
 - an object creating unit for creating an object at the starting process of said object pool according to said object information stored by said object information storage unit;
 - and
 - an object managing unit for pooling the object created by said object creating unit before accessing said object from said client.

Jeffords does not disclose “an object information storage unit for storing object information at the termination process of said object pool”, and also does not disclose “an object creating unit for creating an object at the starting process of said object pool according to said object information stored by said object information storage unit” as recited in claim 3. Furthermore, the reference does not disclose “an object managing unit for pooling the object created by said object creating unit before accessing said object from said client” as also recited in claim 3.

In rejecting the claim, the Examiner refers to col. 14, line 36 to col. 15, line 17 reproduced above as disclosing the subject matter of claim 3. As discussed in detail above, however, Jeffords discloses a method that is primarily directed to establishing contact with other systems. Jeffords does not disclose storing object information at a termination process of an object pool, creating an object at a starting process of an object pool, and pooling the created object before accessing the object from a client.

Claim 3, accordingly, is also not anticipated by Jeffords, and should be allowable thereover in its present form.

Claim 4 depends from and further restricts claim 3 and is also not anticipated by Jeffords, at least by virtue of its dependency.

A.3. Claims 5-7, 10, 12 and 13

Independent claims 5, 10 and 12 recite limitations generally similar to claim 1, and are not anticipated by Jeffords for substantially the same reasons as discussed above with respect to claim 1.

Claims 6 and 7 depend from and further restrict claim 5, and claim 13 depends from and further restricts claim 12, and these claims are also not anticipated by Jeffords, at least by virtue of their dependency.

A.4. Claims 8, 9 and 11

Independent claims 8 and 11 recite limitations generally similar to claim 3, and are not anticipated by Jeffords for similar reasons as discussed above with respect to claim 3.

Claim 9 depends from and further restricts claim 8, and is also not anticipated by Jeffords, at least by virtue of its dependency.

A.5. Claims 14 and 15

Independent claim 15 reads as follows:

15. A program sending apparatus, comprising:
- a storage unit for storing a program which makes a computer execute an object pooling process for pooling, on a server, objects associated with execution of an application utilizing distributed objects, an information storing process for storing object information in said server, and a creation sequence determining process for determining a sequence of objects to be created according to said object information stored by said information storing process; and
 - a sending unit for reading out said program from said storage unit, and sending said program.

Jeffords does not disclose, in col. 14, line 36 to col. 15, line 17 or elsewhere, "a creation sequence determining process for determining a sequence of objects to be created according to said object information stored by said information storing process". Jeffords nowhere discusses determining a sequence of objects to be created and nowhere discloses that such a sequence is according to object information stored by an information storing process.

In responding to Appellants' arguments regarding claim 15 in the Response to Office Action dated June 28, 2004, the Examiner states, in part:

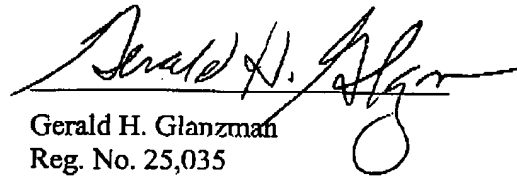
Jeffords states that when initialized the RRM object examines/uses the resource objects in the resource pools on col. 5, lines 37-4. This plurality of objects can be considered a sequence. These resource objects are stored in the distributed memory space by some form of information storing process.

Final Office Action dated November 3, 2004, page 5.

Appellants respectfully disagree. Appellants submit that a disclosure that the RRM is able to examine/use resource objects in a resource pool, does not constitute a teaching of "a creation sequence determining process for determining a sequence of objects to be created according to said object information stored by said information storing process" as recited in claim 15. Accordingly, claim 15 is not anticipated by Jeffords and is believed to be allowable thereover in its present form.

Independent claim 14 recites limitations that are generally similar to claim 15, and is also not anticipated by Jeffords for substantially the same reasons as discussed above with respect to claim 15.

For all the above reasons, claims 1-15 are believed to patentably distinguish over Jeffords, and it is respectfully requested that the Board so find and reverse the Examiner's Final Rejection of the claims.



Gerald H. Glanzman
Reg. No. 25,035
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal is:

1. A client server system using distributed objects, comprising:
a client connected to a communication network for performing an access request to an object;
an application server for performing an application by an actual object according to the access request by said client; and
an object pool server connected to said client through said communication network and connected to said application server for pooling a proxy object corresponding to said actual object and for holding actual object management information that is information on said actual object, wherein said application server notifies said object pool server of an event according to a change in status of said application, and said object pool server automatically updates said actual object management information according to the notification of said event from said application server.
2. The client server system as set forth in Claim 1, wherein the event notified from said application server is formed according to at least one of the result of a process of starting a project and the result of a process of stopping the project.
3. An object pool using distributed objects, comprising:
a client request analyzing unit for analyzing an access request to an object;

an object information storage unit for storing object information at the termination process of said object pool;

an object creating unit for creating an object at the starting process of said object pool according to said object information stored by said object information storage unit; and

an object managing unit for pooling the object created by said object creating unit before accessing said object from said client.

4. The object pool as set forth in Claim 3, wherein the object information stored by said object information storage unit is constructed so that it can be at least recognized to be the last accessed object, and said object creating unit starts creation from said last accessed object.

5. A client server system using distributed objects, comprising:

an object pool connected through a communication network to a client which sends an access request to an object, and for pooling objects and managing object information; and

an application execution environment connected to said object pool for executing an application according to the access request of said client, and notifying said object pool of an event according to a status change of said application, wherein said object pool updates said object information according to the event notification from said application execution environment.

6. The client server system as set forth in Claim 5, wherein an object pool server having the function of said object pool and an application server in said application execution environment are connected to each other through a network or the like, said object pool server pooling objects as proxy objects.
7. The client server system as set forth in Claim 5, wherein said object pool and said application execution environment are formed on the same server.
8. An object pooling method for pooling objects in advance on a specified server to execute an application in a distributed system, said object pooling method comprising the steps of:
 - storing object information when a process of said server is terminated; and
 - creating objects according to said object information when a process of said server is started; and
 - pooling the created objects.
9. The object pooling method of Claim 8, wherein said object information is stored with a predetermined priority, and said objects are created in descending order with respect to said priority.
10. An object pooling method on an object pool which is connected through a communication network to a client which sends an access request to an object, and is connected to an application execution environment in which an actual object is executed, comprising the steps of:

pooling a proxy object corresponding to an actual object in said application execution environment;

recognizing a change in status of the actual object in said application execution environment; and

updating actual object management information according to said recognized status change.

11. A storage medium in which a program to be executed by a computer is stored so that it can be read by an input unit of said computer, wherein said program makes said computer execute:

a monitoring process for monitoring execution status of a project which executes an application utilizing distributed objects;

an event creating process for creating an event according to execution status of the project monitored by said monitoring process; and

an event issuing process for issuing the event created by said event creating process to an object pool server connected through a network or the like.

12. A storage medium in which a program to be executed by a computer is stored so that it can be read by an input unit of said computer, said computer being connected through a network or the like to an application server for executing an application utilizing distributed objects, and to a client which requests the execution of said application, wherein said program makes said computer execute:

an object pooling process for pooling the proxy objects corresponding to actual objects to be executed by said application server; and

an updating process for receiving an event issued from said application server, and updating management information on said actual object according to said event.

13. The storage medium as set forth in Claim 12, wherein said computer is made to further execute:

an execution status managing process for keeping track of execution statuses of said objects, and managing them for each object;

a request analyzing process for analyzing a request including an object creation request and/or an object deletion request from said client; and

a process for executing, upon receipt of a result of a request analysis by said request analyzing process, creation of an object and/or the deletion of an object according to a management result by said execution status managing process.

14. A program sending apparatus, comprising:

a storage unit for storing a software product which makes a computer execute an event forming program for forming an event according to a change in status of an application utilizing distributed objects, and an object pooling program for pooling objects according to the event formed by said event forming process; and

a sending unit for reading out said program from said storage unit, and sending said software product.

15. A program sending apparatus, comprising:

a storage unit for storing a program which makes a computer execute an object pooling process for pooling, on a server, objects associated with execution of an application utilizing distributed objects, an information storing process for storing object information in said server, and a creation sequence determining process for determining a sequence of objects to be created according to said object information stored by said information storing process; and

a sending unit for reading out said program from said storage unit, and sending said program.

EVIDENCE APPENDIX

There is no evidence to be presented.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.